INFO-F404, Operating Systems II

Joël GOOSSENS

Université libre de Bruxelles

Low-power scheduling

C) Low-power scheduling

Outline

- Motivations
- Introduction to hardware aspects
- Introduction to software aspects
- Popular techniques
- Conclusion

Motivations/Applications I

- On PDAs, smartphones and laptops, saving battery life means creating less hazardous waste for the same level of functionality.

Motivations/Applications II

- Medical implant, pacemakers, controller can use the energy of our muscles.
- Artificial retinas (prototypes) use incidental light as an energy source.

Motivations/Applications III

- In aerospace. For instance, the Mars Odyssey spacecraft was designed to use less battery power and smaller solar panels.

- More generally, low power consumption implies less heat dissipation.

CMOS circuit power consumption

- $P_{tot} = P_{static} + P_{dyn}$
- Static currents, leakage (due to transistor imperfections)
- Dynamic currents (circuit switching)
- The following can be used as an approximation of the dynamic power:

$$P_{dyn} \approx \alpha \cdot f \cdot C \cdot V_{dd}^2$$

- where
    - f is the clock frequency
    - α is the activity rate (related to switches)
    - C is the capacity
    - V_{dd} is the supply voltage
    - $f \approx V_{dd}^{(\gamma-1)}$ is the switching frequency

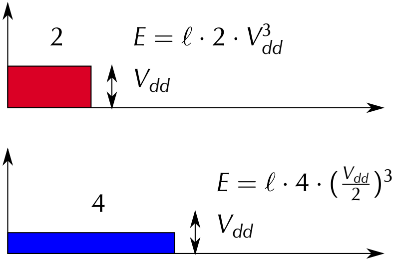
Abstraction

We will assume that $\gamma \approx 2$ as well as the following:

- Speed/computational capacity $\propto V_{dd}$
- Power consumption $\propto V_{dd}^3$

Energy

We'll focus on energy consumption: $E = \int_0^t P(t)\mathrm{d}t$



$$E = \ell \cdot 2 \cdot V_{dd}^3$$

$$E = \ell \cdot 4 \cdot \left(\frac{V_{dd}}{2}\right)^3 = \ell \cdot \frac{V_{dd}^3}{2}$$

Real-time systems

- Real-time systems are systems whose reliability not only depends on the correctness of computations but also on the **instants** when the results are produced. Typically, each computation must be completed while meeting a **deadline**.

Modeling real-time applications

Definition 1 (Periodic tasks)

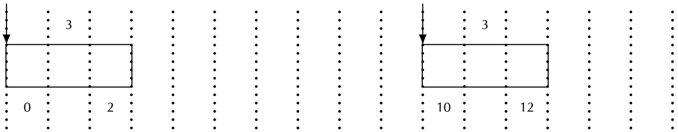
A **periodic task** τ_i is characterized by a tuple (T_i, C_i), where

- C_i is its **worst-case execution time** which specifies an upper bound on the execution time of each job generated by τ_i.
- T_i is the **period** specifying the interval of time separating two consecutive requests of task τ_i.
- We assume implicit deadlines. The absolute deadline of a job is exactly the arrival time of the next job of its task.

Note: C_i may be a number of processor cycles (absolute referential) or a duration based on a reference processor frequency (relative referential).

Periodic task example

$$\tau_1 = (T_1 = 10, C_1 = 3)$$



System load

Definition 2 (Utilization)

The **utilization** $U_i(f)$ of a task τ_i is the ratio between its worst-case execution time and its period: $U_i(f) \stackrel{\text{def}}{=} C_i(f)/T_i$.

The utilization of a periodic task system is the sum of the utilizations of all tasks: $U(f) \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau} U_i(f)$

Scheduling

- **Scheduling** is the computer science term denoting the decision process of choosing which **process** should be run.
- The operating system regularly executes an algorithm called a **scheduler** that chooses which process should be run.
- If the process chosen is not the same as that which ran previously, we must proceed with a context switch (also called **preemption**).

Time-sharing vs. real-time

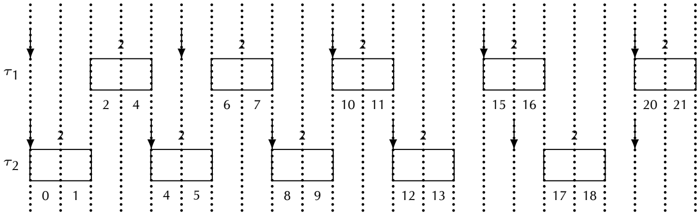
- Schedulers using **time-sharing** are present on most **general-purpose** computer systems.
- **Real-time** schedulers can **guarantee** that jobs get completed **in time**.

Earliest Deadline First — EDF

Definition 3 (EDF)

At each instant, EDF gives highest priority to the active job whose absolute deadline is smallest (i.e. closest).

The following example showcases EDF used on a task system with two tasks: $\tau_1 = (T_1 = 2, C_1 = 5)$, $\tau_2 = (T_2 = 2, C_2 = 4)$.



Assumptions

- EDF scheduler
- preemptive system
- negligible preemption delays
- uniprocessor platform

An offline technique I

This first technique uses the following condition as basis:

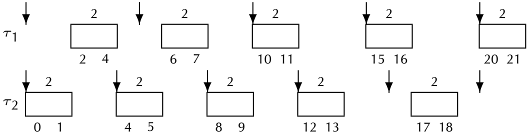
Condition 4

A system is EDF-schedulable at frequency f if and only if

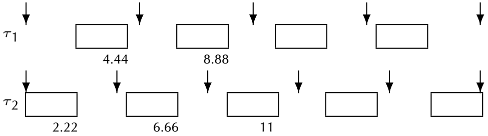
$$U(f) \leqslant 1$$

- The basic idea will be to work with the smallest frequency f such that $U(f) \leqslant 1$ still holds.
- For instance, for the task system made of two tasks $\tau_1 = (T_1 = 2, C_1 = 5)$ and $\tau_2 = (T_2 = 2, C_2 = 4)$, we have $U(1) = 0.9$. Therefore, we could work with a frequency of 0.9, which amounts to multiplying execution times by $1.111\ldots$
- $U(0.9) = 1$

An offline technique II



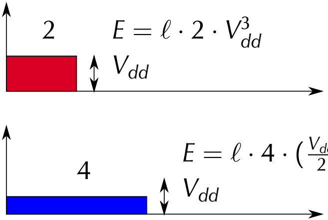
$U(1) = 0.9$



$U(0.9) = 1$

Reminder

Note that the quadratic relation between frequency and power consumption implies that we're saving energy!



$$E = \ell \cdot 2 \cdot V_{dd}^3$$

$$E = \ell \cdot 4 \cdot (\frac{V_{dd}}{2})^3 = \ell \cdot \frac{V_{dd}^3}{2}$$

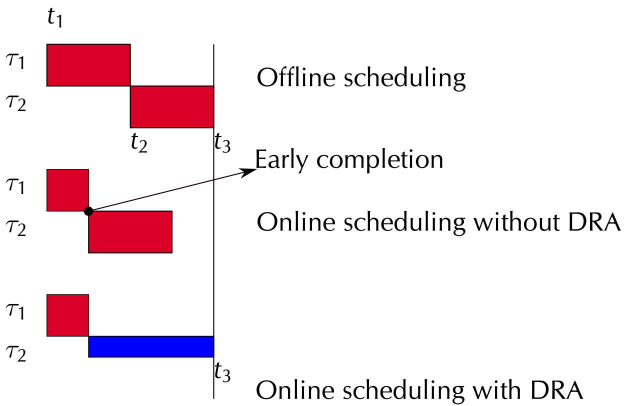
DVFS online techniques — Generalities

- Offline techniques are based on the worst-case execution times (WCET).
- Online techniques use data collected during system execution.
- Frequencies may be adjusted during system execution.
- Dynamic Voltage Frequency Scaling (DVFS) technology

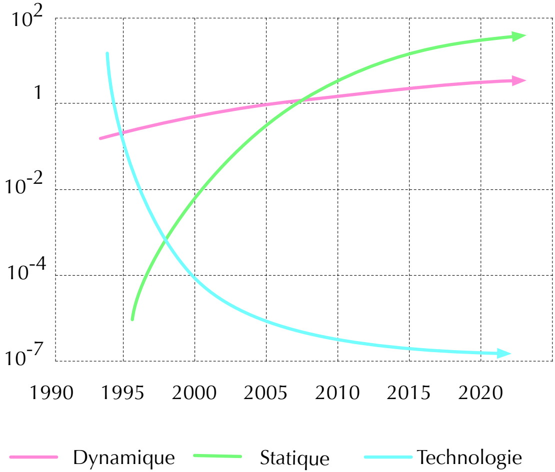
Dynamic Reclaiming Algorithm — DRA [1]

- The DRA technique tries to exploit early job completions.
- Part of the offline schedule is used to meet deadlines, but with the ability to scale down frequency/voltage at times.

DRA — Illustration



Evolution of dissipated power



Power-saving in recent embedded processors

- Hardware mechanisms to control leakage current.
- Used during sufficiently long idle periods.
- Deactivation of **cache blocks**
- Handling of power-saving modes: **freeze**, **hibernation**, **cryo** modes.
- Recent processors have automatic mechanisms that activate during **significantly long** idle periods.

LC-EDF — Leakage-Control EDF [2]

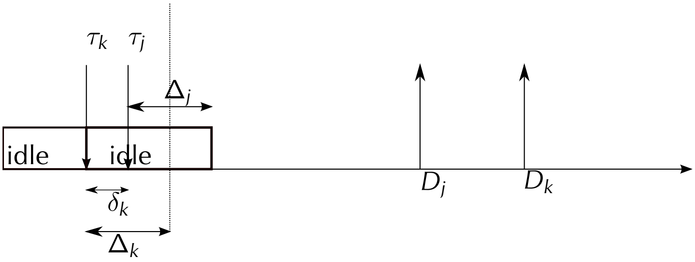
- During an idle period of the processor, we'll attempt to postpone job executions as far as possible.

$$\sum_{i=1, i \neq k}^{n} \frac{C_i}{T_i} + \frac{C_k + \Delta_k}{T_k} = 1$$

LC-EDF

- The same principle applies to later requests of higher priority.

$$\sum_{i=1, i\neq k,j}^{n} \frac{C_i}{T_i} + \frac{C_k + \delta_k}{T_k} + \frac{C_j + \Delta_j}{T_j} = 1$$



Recent research: multiprocessor platforms

- We are reaching the (physical) limits of uniprocessor platforms with regards to clock frequency and integrated circuit density.
- Increasing computational power implies having to use parallel computing, multiprocessor or multicore platforms.

Potential power-saving through parallelism I

As a quick reminder:

- Speed/computational power $\propto V_{dd}$
- Power consumption $\propto V_{dd}^3$

A platform with m processors gives the same computational power if each processors works at a "speed" of $\frac{V_{dd}}{m}$. However, power consumption becomes:

$$m \cdot \left(\frac{V_{dd}}{m}\right)^3 = \frac{V_{dd}^3}{m^2}$$

We gain a factor of m^2!

Potential power-saving through parallelism II

$\lim_{m\to\infty}$ énergie nulle

However, real-time jobs are **sequential**.

"While increasing the number of processors results in lower energy consumption for a given computing capacity, the fraction of that capacity that is guaranteed available for executing the real-time workload decreases as the number of processors increases."

—S. Baruah & J. Anderson
International Journal of Embedded Systems, 2008.

Conclusion

- We have demystified power-saving in digital critical systems.
- We have illustrated the pivotal role of computer science research in reducing production of toxic waste and heat dissipation of electronic equipment.
- We have stressed the new challenges that arise in parallel embedded systems.
- There is a large place for new fresh ideas and technological innovations.

Questions

References I

[1] AYDIN, H., MELHEM, R., MOSSÉ, D., AND MEJÍA-ALVAREZ, P.
Power-aware scheduling for periodic real-time tasks.
IEEE Transactions on Computers 53, 5 (2004), 584–600.

[2] LEE, Y., REDDY, K., AND KRISHNA, C.
Scheduling techniques for reducing leakage power in hard real-time systems.
In *15th Euromicro Conference on Real-Time Systems* (2003), IEEE, pp. 105–112.